Cocoa Text

Session 427

Cocoa Text

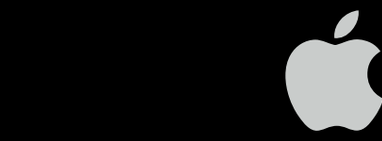
Douglas Davidson
Application Frameworks Engineer

Introduction

- Text system is at the heart of Cocoa
- Responsible for almost all visible text
- Handles everything from simple string display to the fifty-line text editor, and beyond
- Fully Unicode based
- Highly customizable
  - Most of this talk will be about customization

Features

- Full Unicode text display and editing
- Cut/copy/paste, drag and drop, undo/redo
- Rich text with fonts, colors, underlines, sub/superscript
- Paragraph styles with margins, tab stops, line spacing
- In-line attachments of files or graphics
- Reads/writes plain text, RTF, and RTFD, reads HTML
- Pluggable spellchecking



Demo

New Features

- Outlines, shadows, underlines, strikethroughs
- Tooltip and cursor attributes, link attributes
- New paragraph style features
- New glyph generator and typesetter APIs
- New font panel and text ruler features
- Built-in find panel and word completion
- Read/write Word document formats

Cocoa Text and WebKit

- WebKit for display of web pages
  - Most web pages today are not much like text pages
  - Frames, Java, JavaScript, plugins, etc.
- Cocoa text system for display and editing of text
  - Optimized for editing
  - Highly customizable
  - HTML import feature most useful for text-like HTML

Text System Classes

- Model—View—Controller
- Model classes model characters, attributes, geometry (NSTextStorage, NSFont, NSTextContainer)
- Controller classes control glyph generation and layout (NSLayoutManager, NSGlyphGenerator, NSTypesetter)
- View classes handle user input and display (NSTextView)

View Classes

- NSTextView handles display and input for a single NSTextContainer region
- Text view is the face of the text system for most UIs
- Multiple text views on the same document work together
- Ruler classes handle text rulers
- NSTextAttachmentCell draws attachment

New View Features

- Updated font panel
- Updated text ruler
- New find panel
- Completion
- Cursors and tooltips

Model Classes

- Document is NSTextStorage, a mutable attributed string (Unicode characters plus attributes)
- Attributes named by strings, standard or custom
- NSFont, NSColor, NSParagraphStyle, NSShadow
- NSTextAttachment models attached file
- NSTextContainer models geometry of layout

Text Attributes

- Standard attributes are listed in AppKit header
- Font, colors, underlines, strikethroughs, sub/superscript, ligatures, kerning, baseline offset
- Bold/italic, etc., are traits of font, see NSFontManager
- Margins, tab stops, line spacing, paragraph spacing are part of paragraph style
- Cursor, tooltip, link, attachment attributes

New Attribute Features

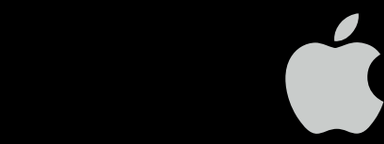
- Stroke width, stroke color
- Underline style, underline color
- Strikethrough style, strikethrough color
- Shadow
- Obliqueness, expansion
- Cursor
- Tooltip

Customization, Part I

- Working with NSTextView
- Notification and delegation before subclassing
- Text view's delegate has extensive control over the handling of user input
- Subclass for more detailed control of other UI features (completion, context menus, copy/paste)
- Subclass for additional drawing over, under, or around the text on the page

Customization, Part I

- Working with the text
- User-initiated text changes go through text view
- Programmatic text changes go through text storage
- Treat text storage as mutable attributed string
- Wrap sequence of changes in beginEditing/endEditing
- Subclass for detailed control of text/attribute storage



Demo

Text View Customization

Characters and Glyphs

- Characters are Unicode code points
- Glyphs are displayable elements of a specific font
- Text system converts characters plus attributes into glyphs plus locations
- One character may produce multiple glyphs, or multiple characters may be represented by one glyph

Example

- Character: Latin small letter a acute (á)
- Attributes: Helvetica Regular 64, blue, centered
- Glyphs: a ´
- Displayed result:

á

Controller Classes

- NSLayoutManager is the boss
- Notified of changes by text storage
- Performs glyph generation and layout on demand
- Calls on NSGlyphGenerator and NSTypesetter
- Stores information about glyphs and layout
- Displays glyphs when requested by view

How Text Gets On the Screen

- Attribute fixing: text storage ensures attributes are OK
- Glyph generation: glyph generator converts characters plus fonts into glyphs
- Layout: typesetter positions glyphs in lines on the page
- Display: layout manager sends glyphs to Quartz
- All of these are usually done lazily and on demand

Attribute Fixing

- Attribute fixing is performed on demand by text storage
- Font fixing substitutes fonts so the text can be displayed
- Paragraph fixing makes sure the whole paragraph has the same paragraph style
- Attachment fixing makes attachment attributes go with attachment characters
- Can be customized in NSTextStorage subclass

Glyph Generation

- Layout manager observes changes to text storage and invalidates as necessary (can also be done manually)
- Glyph generation and layout performed on demand
- Glyph generator converts characters plus fonts to glyphs
- Layout manager stores glyphs, attributes, and indexes
  - Control glyph for tab stops, line breaks, etc.
  - Null glyph sometimes used for padding

Layout

- Layout manager asks typesetter to perform layout
- Typesetter contacts text container for geometry
- Typesetter may insert, change, or hide glyphs
- Typesetter tells layout manager which container a glyph goes in, which line fragment, and where in the line
- Layout manager stores positions of glyphs

Display

- Layout manager performs display at view's request
- Uses Quartz to draw backgrounds, underlines, glyphs
- NSTextView does it, your view can too
- Must have locked focus on a flipped view
- Positions of fragments are relative to text container, glyphs relative to fragment

Customization, Part II

- Talk to the layout manager for information about the layout
- Glyph range for character range and vice versa
- Glyph range for text container and vice versa
- Position, size, and glyph range for line fragment (position in container coordinates)
- Line fragment and position in fragment for glyph
- Glyph for coordinates, bounding rect for glyph range, selection rect array for glyph range

Customization, Part II

- Subclass layout manager to provide custom drawing at the glyph level
- Layout manager draws glyphs, underlines, backgrounds
- Subclass to store additional glyph attributes
- Subclass to override hit testing, selection rectangles, etc.

Demo

Layout Manager Subclass

NSTypesetter

- NSTypesetter is abstract class
- NSATSTypesetter is newly public concrete subclass
- Subclass NSTypesetter for complete custom layout
- Subclass NSATSTypesetter for customization
- Call NSATSTypesetter directly for low-level layout

NSTypesetter Interface

- Layout manager asks typesetter to lay out lines
- Typesetter gets text from text storage, glyphs from layout manager, geometry from text containers
- Typesetter calls back to layout manager with container, fragment, and position information for glyphs
- Typesetter may also insert and modify glyphs and glyph attributes (e.g., hyphens, ellipses, swashes)

NSATSTypesetter Interfaces

- Subclass interface for customization of layout
- Layout phase interface specifically for subclassers
- Glyph storage interface allows use of typesetter without layout manager or text containers
- External interface used by layout manager, or by other clients if glyph storage interface overridden

Glyph Storage Interface

- Encapsulates all typesetter calls back to layout manager
- Subclasses can override with custom implementation
- Allows use of NSATSTypesetter with some custom method of glyph storage rather than NSLayoutManager
- Permits NSATSTypesetter instance to be invoked in isolation, without layout manager or text containers
- Gives low-level access to system line layout algorithms

Customization, Part III

- Subclass NSATSTypesetter to customize layout
- Be notified of or modify line fragments
- Control line breaking and hyphenation
- Control behavior of tabs and other control glyphs

Demo

Typesetter Subclass

Wrap Up

Roadmap

400 Introduction to Developing Applications with Cocoa	Nob Hill Mon., 5:00pm
410 Cocoa Update	Presidio Wed., 10:30am
412 What's New in Cocoa User Interface Programming	Presidio Wed., 3:30pm
413 Cocoa 2D Drawing Techniques	Nob Hill Wed., 5:00pm

Roadmap

419 What's New in Cocoa User Interface Programming In Depth	Russian Hill Fri., 9:00am
424 Cocoa: Tips and Tricks	Mission Fri., 2:00pm
403 Internet Technologies: Safari Overview	Presidio Tue., 2:00pm
404 Unicode for Japanese, Chinese and Everything Else	Russian Hill Tue., 2:00pm
406 What You Need to Know about Fonts in Mac OS X	Mission Tue., 3:30pm

Who to Contact

John Geleynse
Manager, Software Evangelism
geleynse@apple.com

Cocoa Feedback
For suggestions and feedback
cocoa-feedback@group.apple.com

Developer Technical Support
Code-level questions (fee-based)
dts@apple.com

http://developer.apple.com/wwdc2003/urls.html

For More Information

Reference Library

Documentation

Application Kit Reference Documentation

ADC Home > Documentation > Cocoa >

Text Handling Programming Topics

ADC Home > Documentation > Cocoa >

Cocoa Text Release Notes

ADC Home > Documentation > ReleaseNotes >

Reference Library

Sample Code

TextEdit

/Developer/Examples/AppKit/TextEdit

TextSizingExample

/Developer/Examples/AppKit/TextSizingExample

CircleView

/Developer/Examples/AppKit/CircleView

Reference Library

Sample Code (Cont.)

LayoutManagerDemo

ADC Home > Sample Code > Cocoa >

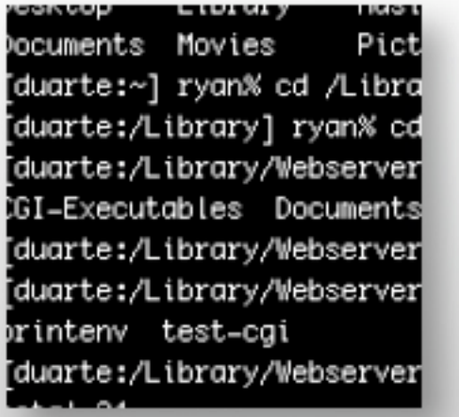
TextViewConfig

ADC Home > Sample Code > Cocoa >

TextViewDelegate

ADC Home > Sample Code > Cocoa >

Q&A



http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference